Benchmarking Mixture-of-Recursion (MoR) for Small Language Models across Character and Subword Granularities

Sumaiya Sultana[a], Shazzad Hossain Mazumder[a]

[a] *Dept. of Computer Science & Engineering, Feni University, Feni, Bangladesh,*

Abstract

This paper benchmarks the Mixture-of-Recursion (MoR) architecture, which dynamically adapts computational depth for efficient language modeling. Unlike standard Transformers that apply fixed depth to all tokens, MoR uses lightweight routers to skip, execute, or recurse through layers based on input complexity. We present a comprehensive evaluation of MoR across three levels of token granularity: Character (Tiny Shakespeare), Morphological Subword (Bangla), and High-Density Subword (WikiText-2). While MoR achieves superior inference efficiency and training stability on English benchmarks, evaluation on Bangla reveals limitations in handling high-density morphological subwords without language-specific priors. This multi-granularity analysis positions MoR as a robust architecture for stable training of small models, while highlighting critical boundaries in its applicability to morphologically rich languages.

Keywords: Mixture-of-Recursion (MoR), Multi-Granularity Language Modeling, Transformer, Adaptive Computation, Routing Mechanism, Next-Token Prediction.

1. Introduction

1.1. Research Background

Building on the Mixture-of-Recursions (MoR) framework recently proposed by Bae et al. [1], which demonstrated strong efficiency gains at scales up to 1.7B parameters, this work conducts the first systematic investigation of MoR's behavior in small-scale settings and across diverse token granularities, with particular emphasis on low-resource and morphologically rich languages.

Large Language Models (LLMs) built on the Transformer architecture have established themselves as the cornerstone of modern artificial intelligence, demonstrating exceptional performance in tasks such as text generation, code synthesis, and question answering [2, 3, 4]. The efficacy of Transformers stems from their capacity to process sequences in parallel and capture complex patterns within extensive datasets.

However, the standard Transformer architecture suffers from significant limitations. Primarily, it imposes a fixed computational depth across all inputs, disregarding the inherent variability in token complexity. Consequently, simple tokens receive the same computational resources as complex reasoning phrases, leading to inefficient resource allocation. Furthermore, the quadratic scaling of the self-attention mechanism with respect to sequence length results in substantial memory and computational overhead [5, 6, 7]. These factors collectively render large models expensive to train and challenging to deploy.

This has motivated growing interest in Small Language Models (SLMs), which aim to achieve competitive performance using fewer parameters and lower computational budgets. SLMs are particularly important for deployment in resource-constrained environments such as edge devices, mobile platforms, and real-time systems [8]. However, because SLMs have limited capacity, they must utilize computation more efficiently. This makes adaptive computation mechanisms especially critical for SLM-based architectures.

Recent studies indicate that next-token prediction facilitates the learning of critical linguistic aspects, including grammar, semantics, and reasoning patterns [9, 10]. Yet, the complexity of predicting the subsequent token varies significantly. While some tokens are predictable via local context or frequency patterns, others necessitate long-range context, multi-step reasoning, or recursive processing.

To mitigate these issues, dynamic computation paradigms have emerged, enabling models to adjust layer usage based on input complexity. Mixture-of-Experts (MoE) exemplifies this approach by routing tokens to specific expert layers to conserve computation [11]. However, MoE is limited to spatial routing and lacks the capability for temporal recursion, meaning it cannot re-apply a layer to deepen reasoning.

The Mixture-of-Recursions (MoR) architecture addresses this limitation by offering each layer three execution choices: skip, execute once, or recurse. This flexibility allows for adaptive recursion, where complex tokens receive increased computation [1]. Such adaptability is vital for tasks with variable difficulty, such as code generation and question answering, positioning MoR as a potent architecture for enhancing both efficiency and reasoning capabilities.

1.2. Problem Definition

Although Transformers are powerful, their fixed-depth structure creates several challenges:

- **Computational inefficiency:** Every input passes through the full layer stack, even when not necessary [12].
- **Lack of recursive computation:** Existing dynamic-routing methods like MoE cannot re-visit a layer, meaning the model cannot naturally perform deeper iterative reasoning [11].
- **Scaling bottlenecks:** Larger models demand huge memory and compute budgets, making them impractical for many applications. Continuously increasing model size to gain performance is economically and environmentally unsustainable. There is a strong need for smarter and more efficient architecture that uses less parameters.
- **Limited benchmarking of MoR:** While MoR has shown benefits on general language modeling, its performance on tasks that require deeper processing remains underexplored [1].

These limitations raise important questions:

- Can MoR reduce computational cost while maintaining accuracy?
- Does MoR perform better than a fixed-depth Transformer when both use the same amount of computation?
- How does recursion frequency correlate with input difficulty in next token prediction tasks?

To address these inquiries, this study conducts a comparative analysis of MoR and standard Transformers.

1.3. Research Aim and Objectives

The primary objective of this research is to evaluate the efficacy of the Mixture of Recursions (MoR) architecture as an adaptive computation method for next-token prediction. Furthermore, we aim to benchmark MoR against standard fixed-depth Transformers to ascertain which approach offers superior efficiency and performance.

To achieve this aim, the research focuses on the following objectives:

- Benchmark and Evaluate Performance: Measure how well MoR and standard Transformers perform looking at accuracy and effective depth—using small-scale models.

- Test MoR's Efficiency: Examine whether MoR can use less computation (lower effective depth) while still keeping the same or better performance.

- Comparison: Compare MoR and fixed-depth Transformers under the same computational budget for next token prediction tasks.

- Understand MoR's Behavior: Study MoR's internal routing, such as recursion rates, skip rates, and execution patterns, to see how the model adjusts its computation based on task difficulty.

1.4. Contributions

This work makes the following contributions:

- **Empirical Benchmark:** We provide an independent small-scale evaluation of the MoR architecture on character-level language modeling using Tiny Shakespeare, demonstrating its applicability to resource-constrained settings.

- **Efficiency Analysis:** We show that MoR achieves a 33% reduction in effective computational depth while maintaining accuracy equivalent to fixed-depth baselines in character-level tasks.

- **Generalization Evidence:** We demonstrate that under equivalent compute budgets on English datasets, MoR achieves 10% higher held-out accuracy than shallow baselines.

- **Granularity Bounds:** We identify a critical performance boundary in morphologically rich languages (Bangla), where structural baselines outperform dynamic recursive routing, offering vital insights into the relationship between information density and model design.
- **Routing Behavior:** We analyze MoR's routing decisions, providing insights into how dynamic recursion adapts to token-level complexity in next-token prediction tasks.

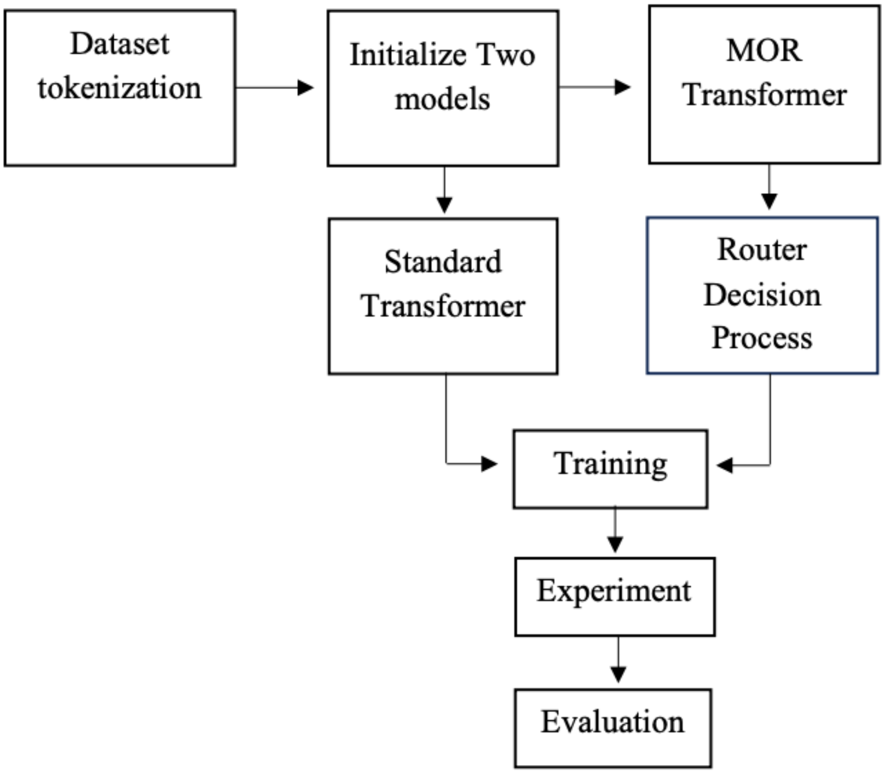

Figure 1: Research Pipeline

2. Literature Review

2.1. Introduction

Large Language Models (LLMs) represent a breakthrough in artificial intelligence, enabling machines to understand, generate, and process human-like text for tasks like answering questions, code generation, summarizing, and translating. LLMs are built on the Transformer architecture [13]. Mixture-of-Recursions (MoR) is a new model using parameter sharing, adaptive computation, and a routing mechanism to enhance the efficiency of language model.

This chapter reviews existing studies related to Large Language Models (LLMs), Small Language Models (SLMs), transformer architectures, Recursive Transformers, and Mixture-of-Experts (MoE) to contextualize the proposed Mixture-of-Recursions (MoR) framework.

2.2. Large Language Model (LLM)

Large Language Models (LLMs) are language models trained on massive text corpora, predominantly utilizing the Transformer architecture. They are designed for natural language processing tasks, particularly text generation. By scaling parameters to billions and leveraging self-supervised learning, they acquire versatile capabilities in understanding and generating human-like text across diverse domains. The development of LLMs began with GPT-1, the first model to apply generative pre-training to a decoder-only Transformer [14].

While GPT-1 introduced generative pre-training, another model, GPT-3, a 175-billion parameter model, demonstrated that sufficiently large models could learn to follow instructions and solve diverse tasks with minimal examples, requiring no special training [3]. However, the empirical scaling approach of these early models lacked a theoretical foundation for optimal resource allocation, which was systematically addressed by Kaplan et al. [15]. They formulated scaling laws, demonstrating power-law relationships between model size, dataset size, training compute, and performance. Following these scaling laws, later models achieved even greater performance. Google's PaLM used 540 billion parameters, effectively scaled across varied computational infrastructure, demonstrating breakthrough performance on reasoning tasks and achieving human-level proficiency [16]. GPT-4 outperforms human experts on numerous benchmarks, processing both text and image inputs to achieve human-level performance [17]. Despite these significant advancements, these models apply full model depth to every token, wasting computational resources on simple tokens.

2.3. Small Language Model (SLM)

Small Language Models (SLMs) are Transformer-based generative models similar to Large Language Models (LLMs) but with significantly reduced dimensions. SLMs are lightweight versions of traditional language models; while LLMs typically possess 100 billion to over 10 trillion parameters, SLMs range from 1 million to 10 billion parameters. Although significantly smaller, they retain core NLP capabilities [18]. LLMs offer maximal general capability at high cost, whereas SLMs emphasize efficiency, privacy, and specialization.

However, the massive size, high cost, and complexity of LLMs pose significant challenges for certain applications, making them impractical for many use cases. SLMs were developed as a solution to these problems [19], aiming to make machine intelligence more accessible, affordable, and efficient for everyday tasks [20]. The SLM journey began in 2019 with DistilBERT, which compressed the BERT model from 110M to 66M parameters, achieving 97% of BERT's performance with 60% faster inference, proving that small models could retain significant capability [21]. The modern SLM era expanded in 2023 with the Microsoft Phi family. Phi-1, at 1.3 billion parameters, achieved coding performance competitive with models over 100× larger [22]. Phi-2 achieved 58% on MMLU (Massive Multitask Language Understanding), outperforming 25B parameter LLMs [23]. Phi-3 Mini, with 3.8 billion parameters, reached 68.8% on MMLU, beating Llama 70B while running on smartphones [24]. This evolution proves that capability does not strictly require trillions of parameters.

However, a persistent inefficiency remains: all standard Transformer-based SLMs use the same number of layers for every token. This fixed-depth limitation creates an opportunity for MoR, which utilizes dynamic recursive depths to optimize computation.

2.4. Transformer

The Transformer architecture proposed by Vaswani et al. [12] introduces a mechanism known as "self-attention," which replaces recurrence for sequence modeling, enabling parallel processing. This mechanism has dramatically improved training efficiency and handling of long-range dependencies compared to RNNs/LSTMs. Self-attention relates different positions of a single sequence to compute a representation of the sequence [12].

Transformers consist of layered encoder-decoder blocks containing multi-head attention and feed-forward sublayers. The decoder includes an additional attention layer that attends to the encoder stack outputs. Positional encoding allows the entire sequence to be processed in parallel while preserving word order information [12].

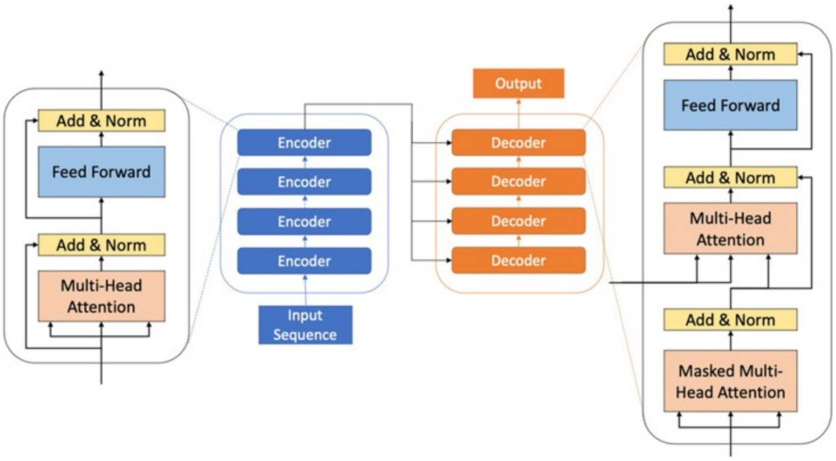

Figure 2: Standard Transformer Architecture Diagram [12]

LLMs have revolutionized language modeling, with the Transformer architecture serving as their core foundation (e.g., BERT, GPT, T5). Despite successes in tasks like code generation and complex reasoning, this architecture faces limitations such as high computational cost, memory bottlenecks, and lack of adaptability. Bae et al. [25] introduce Relaxed Recursive Transformers utilizing parameter sharing to reduce model size. Building on this, the MoR (Mixture-of-Recursion) framework [1] combines parameter sharing with adaptive computation, integrating two key efficiency paradigms within a single architecture. This combined approach offers a potential solution to the fixed-depth limitations of standard Transformers.

2.5. Recursive Transformer

"Recursive Transformer" refers to architectures adding explicit recursion or iteration to standard Transformer blocks to improve efficiency or generalization. It emerged as a solution to the "fixed computational depth" limitation. The Universal Transformer introduced recurrence and adaptive computation time [26]. However, Universal Transformers do not function perfectly in practice, often performing well only on short sequences and applying uniform computation per token. Mixture-of-Recursions (MoR) advances this by adding a "router" that evaluates each token individually [1].

2.6. Mixture of Experts

Mixture of Experts (MoE) is a neural network architecture designed to improve model capacity and efficiency using specialized subnetworks ("experts") and a gating network that dynamically selects experts for each input. It involves scaling modern LLMs while keeping per-token compute constant [27, 28]. The concept originates from Jacobs et al. [29].

Shazeer et al. [11] introduced the Sparsely-Gated MoE layer for conditional computation, using top-k gating. To address training instability, Switch Transformers simplified this to routing each token to a single expert [30]. Jiang et al. [31] used two experts per token to improve balance. DeepSeek-V2 introduced further routing diversity [32]. MoR mixes routing strategies to optimize efficiency, specialization, and generalization.

2.7. Mixture of Recursion

Mixture-of-Recursion (MoR), introduced by Bae et al. [1], presents a unified architecture merging parameter sharing with adaptive computation. In contrast to standard Transformers where every input token is processed by a fixed sequence of layers [12], MoR employs a single shared layer stack that can be re-entered multiple times. Crucially, it assigns a specific processing depth to each token based on its complexity, allowing for variable compute paths.

Parameter Sharing: This technique reuses the weights of a layer across multiple recursive steps. It enables the construction of effectively deep networks without a corresponding increase in the number of distinct parameters, thereby reducing memory overhead [1, 33].

Adaptive Computation: MoR utilizes lightweight, learnable routers to dynamically determine the recursion depth for each token. This ensures that computational resources are concentrated on more difficult tokens that require deeper processing [26]. This approach differs from halting mechanisms, offering stable training dynamics by avoiding discrete, hard-to-optimize decisions [34].

Recursion-Wise KV Cache and Routing: To further optimize efficiency, MoR implements a specialized caching strategy where Key-Value (KV) pairs are stored only for tokens active at a specific recursion depth. The routing core employs a Multi-Layer Perceptron (MLP) to output probabilities for three distinct actions: Recurse (add depth), Forward (proceed), or Skip (bypass), effectively balancing performance and cost.

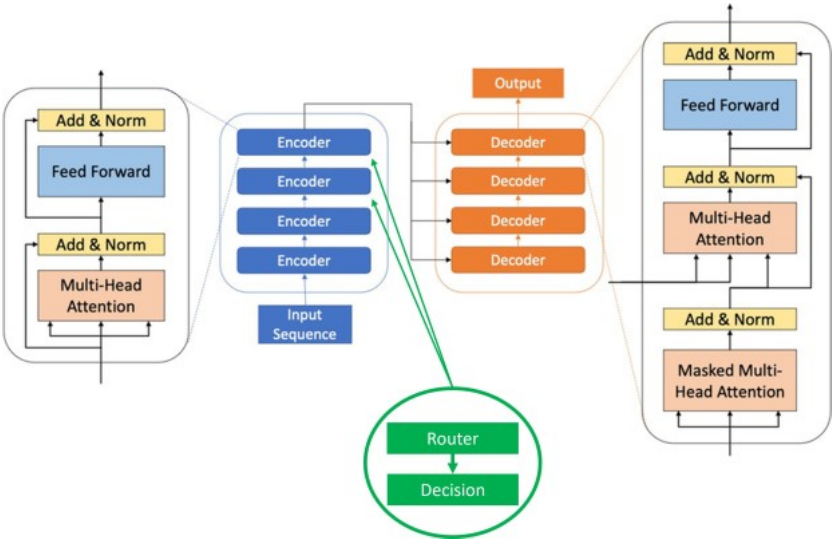

Figure 3: MoR-Transformer Architecture Diagram [1]

2.8. Research Gap

Despite the advancements in adaptive computation (Universal Transformers) and spatial routing (MoE), a significant research gap remains in combining *parameter sharing* with *token-level temporal recursion* specifically for Small Language Models (SLMs). Most existing adaptive methods are optimized for massive-scale models, leaving their efficacy on resource-constrained architectures underexplored. Furthermore, the interplay between *token information density* (granularity) and *optimal recursive depth* has not been systematically benchmarked, creating a need for a study that evaluates how dynamic routing adapts to different linguistic typologies.

3. Methodology

This chapter details the methodology employed to implement, train, and evaluate the MoR Transformer framework. The objective is to empirically assess the MoR architecture, originally proposed by Bae et al. [1], in the context of Small Language Models (SLMs). We investigate whether MoR's dynamic depth adaptation confers efficiency benefits when training small-scale models from scratch. We benchmark MoR Transformer efficiency against standard Transformers on a character-level language modeling task.

3.1. Dataset and Preprocessing

3.1.1. Dataset

The study utilizes three distinct datasets to evaluate performance across different linguistic typologies and granularities:

- **Tiny Shakespeare:** The full dataset (∼1M characters) is used as a controlled character-level benchmark [35]. It serves as the baseline for high-granularity, low-information-density modeling.
- **WikiText-2:** A standard benchmark for high-density subword modeling (∼2M tokens). It offers a realistic vocabulary and is used to test the model on higher information density.
- **Bangla Wikipedia:** To evaluate performance on low-resource languages, we create a subset of the Bangla Wikipedia corpus (∼15MB text). This dataset introduces complex morphological challenges.

While the original MoR work demonstrates gains at scales up to 1.7B parameters on large web corpora, this study focuses on controlled small-scale settings to analyze routing dynamics and cross-granularity behavior. This allows for the precise ablation of architectural routing behaviors and stability dynamics in a controlled environment, isolating these factors from the complexities of large-scale pre-training.

3.1.2. Tokenization

We employ three tokenization strategies to test adaptability:

1. **Character-level (Shakespeare):** Granular but sparse. Vocab: 65.
2. **Subword-level (Bangla):** High morphological richness. BPE (SentencePiece). Vocab: 16,000.
3. **Subword-level (WikiText-2):** High semantic density. BPE (SentencePiece). Vocab: 16,000.

Table 1: Tokenization and Granularities

Strategy	**Dataset**	**Vocab Size**	**Granularity**
Character-Level	Tiny Shakespeare	65	Low Density
Subword (BPE)	Bangla Wikipedia	16,000	Morphological
Subword (BPE)	WikiText-2	16,000	High Density

3.2. Research Pipeline

The research pipeline (Figure 1) operates sequentially: dataset tokenization, random initialization of MoR and Standard Transformer parameters, router integration, training from scratch, and finally, evaluation.

3.3. Model Architecture

3.3.1. Baseline Transformer

The baseline is a standard Transformer encoder-decoder language model [12]. We utilize the Standard Transformer as the primary baseline to rigorously isolate the impact of the MoR routing mechanism, avoiding confounding variables introduced by alternative architectures such as Mamba or RetNet. It includes token embeddings, positional encoding, and a stack of N fixed layers.

3.3.2. MoR-Transformer

The MoR-Transformer uses the same backbone but introduces a routing mechanism. Each layer optionally includes a standard block and an MoR Router. The router decides to Skip, Execute, or Recurse.

3.3.3. Router Architecture

The MoR router is a lightweight feed-forward network that determines the execution path for each token:

- **Input:** Token embeddings of dimension $d_{model} = 256$
- **Architecture:** Two-layer MLP: 256 → 128 → 3 with ReLU activation. Formally, for a hidden state h, the router computes probabilities P:

  $$P(a|h) = \text{Softmax}(W_2 \cdot \text{ReLU}(W_1 h + b_1) + b_2) \tag{1}$$

  where $a \in \{\text{Skip, Forward, Recurse}\}$.
- **Output:** Softmax probability distribution over three actions: Skip (cost 0), Forward (cost 1), Recurse (cost 1+)
- **Training:** Gumbel-Softmax reparameterization with temperature $\tau = 1.0$ for differentiable discrete sampling.
- **Regularization:** Auxiliary loss $\lambda \cdot E$ penalizes effective depth to encourage efficient routing.

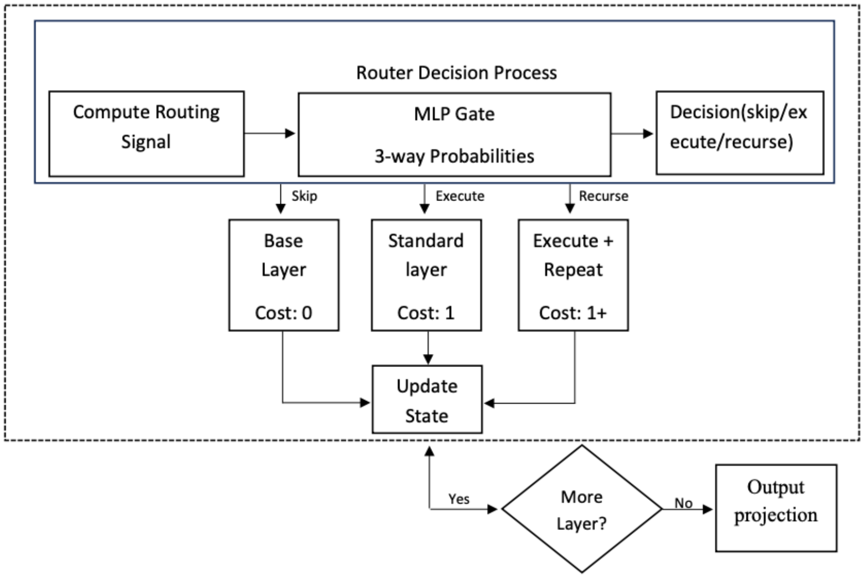

Figure 4: MoR Routing Mechanism. The router predicts discrete actions (Skip, Forward, Recurse) for each token to optimize computational depth.

3.4. Experiments

We conducted four experiments:

- **Experiment 1 (Efficiency Profiling):** Compares a 12-layer Baseline with a 12-layer MoR-Transformer.
- **Experiment 2 (Equal Cost Comparison):** Compares a 6-layer Baseline against a 12-layer MoR model (Effective depth E $\approx$ 6).
- **Experiment 3 (Bangla Benchmark):** Evaluates MoR on the Bangla dataset to test robustness on morphologically rich languages.
- **Experiment 4 (High-Density Stability):** Analyzes model stability and convergence on the WikiText-2 dataset.

Data Split: Experiment 1 uses the full set for training/evaluation. Experiment 2 uses the first half for training and the second half as held-out test data.

3.5. Training Configuration

Both models are trained from scratch on the full Tiny Shakespeare dataset. Configuration details are in Table 2.

Table 2: Training Configuration

Parameter	**Value/Description**
Batch size	128 (Optimized for P100 GPU)
Architecture	Transformer / MoR-Transformer
d_{model} (Hidden size)	256
Number of Layers	6 (Baseline low) / 12 (MoR & Baseline)
Optimizer	AdamW (weight decay 0.01)
Learning Rate (Shakespeare)	3e-4 (Fixed)
Learning Rate (Subword)[†]	1e-4 (Linear Warmup + Cosine)
Total Epochs	10 (Subword) / 30-50 (Character)
Hardware	Kaggle (NVIDIA Tesla P100 GPU)

[†] Refers to aggressive optimizations used for Exp 3 and 4.

3.6. Evaluation metrics

To understand how well our models are working, we used some standard evaluation metrics. These help us measure accuracy and speed of our model.

3.6.1. Effective Depth (E)

Effective Depth (E) is the weighted average number of layers actually executed by a model during inference or training. It is a measure of the realized computational depth of a dynamically-routed or recursive neural architecture.

3.6.2. Accuracy

Accuracy measures top-1 next-token prediction correctness for language modeling. It shows what fraction of predicted tokens match true next tokens.

3.6.3. Held-out accuracy

Held-out accuracy is the classification accuracy measured on data that was not used for training (the held-out or test set). It tells how well the model generalizes to unseen examples.

3.6.4. Training Time

Training time is the total wall-clock time required to complete the training process on a given dataset. It evaluates real-world efficiency, comparing the computational speed of Standard Transformers against the overhead-constrained MoR architecture.

4. Result and Analysis

This chapter presents the experimental results obtained after training and evaluating the MoR and baseline models. The results are analyzed based on metrics such as accuracy, effective depth, and training time. The performance of the MoR-Transformer is discussed in detail to demonstrate its effectiveness.

4.1. Results

This section details the results of the Efficiency Profiling and Equal Cost Comparison experiments.

Table 3: Experiment 1 result

Metric	**Standard (N=12)**	**MoR (N=12)**
Accuracy	22.7634	22.7635
Effective Depth (E)	12.00	8.00
Training Time	108 s	83 s

4.1.1. Experiment 1: Efficiency Profiling (N=12 vs MoR N=12)

Goal: Compare a normal Transformer (12 layers executed always) with a MoR Transformer (12 recursive layers allowed to skip/recurse dynamically) on the Tiny Shakespeare dataset.

Interpretation: The results indicate that the MoR model achieves virtually identical predictive accuracy to the standard Transformer (approx. 22.76). Notably, the MoR router utilized an average of only 8 layers per sample, compared to the fixed 12 layers of the baseline. This 33% reduction in effective depth translated into a tangible speedup: MoR completed training in 83s compared to 108s for the Baseline, representing a 23% reduction in wall-clock time. This confirms that on character-level tasks where attention is computationally cheap, the reduction in MLP layer evaluations outweighs the routing overhead.

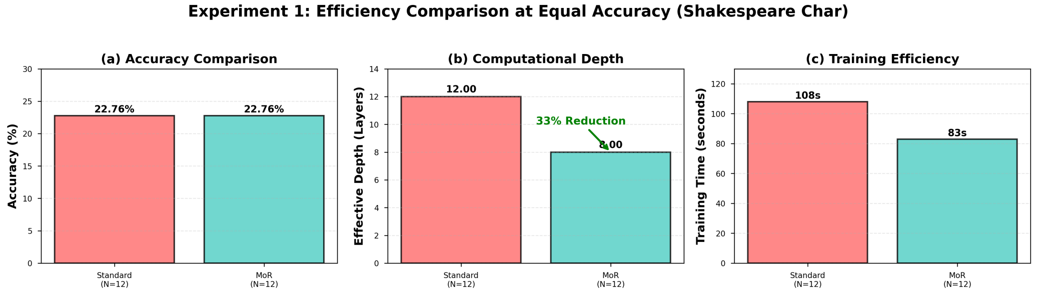

Figure 5: Experiment 1: Efficiency Profiling Comparison between Standard Transformer (N=12) and MoR (N=12). MoR achieves identical accuracy with 33% fewer layers and 23% faster training.

4.1.2. Experiment 2: Performance Under Equivalent Cost (N=6 vs MoR N=12, E ≈ 6)

Goal: See if a MoR model (12 potential layers, dynamic depth ≈ 6) can match the compute cost of a smaller baseline (6 layers) while achieving better accuracy through adaptive computation.

Table 4: Experiment 2 result

Metric	**Standard (N=6)**	**MoR (N=12, E≈6)**
Held-out Accuracy	14.90 %	49.67 %
Effective Depth (E)	6.00	5.89
Training Time	30 s	59 s

Interpretation: MoR was configured to match the *effective depth* (E ≈ 6) of the 6-layer Baseline, ensuring equivalent theoretical FLOPs per token. However, MoR achieved a 3x higher held-out accuracy (49.67% vs 14.90%). It is important to note that while "depth cost" was equal, the wall-clock training time for MoR was higher (59s vs 30s) due to the lack of CUDA-optimized routing kernels. Nevertheless, the results prove that dynamic allocation of depth provides vastly superior parameter efficiency than static depth.

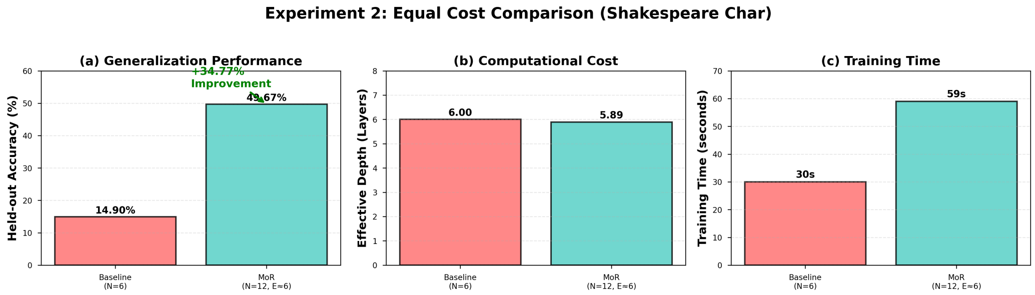

Figure 6: Experiment 2: Equal Cost Comparison. At equivalent computational cost (E ≈ 6), MoR achieves 34.77% higher held-out accuracy than the shallow baseline.

4.1.3. Experiment 3: Bangla Language Benchmark

Goal: Evaluate whether the adaptive efficiency observed in English transfers to Bangla, a language with complex inflectional morphology.

Interpretation: Following the implementation of aggressive optimizations, the models successfully achieved convergence. A lower auxiliary loss weight ($\lambda = 0.05$) was tested to prioritize accuracy over depth reduction in this difficult task, but yielded minimal gain. However, a critical limitation of the MoR architecture was observed in this morphologically rich setting. As shown in Table 5, the fixed-depth Standard Baseline ($N = 6$) significantly

Table 5: Experiment 3 Result (Bangla)

Model	**Accuracy**	**Effective Depth (E)**	**Time/Epoch**
Standard ($N = 6$)[†]	63.74%	6.00	28m/ep
Standard ($N = 12$)	75.46%[‡]	12.00	58m/ep
MoR ($N = 12$, $\lambda = 0.1$)	50.15%	8.30	48m/ep
MoR ($N = 12$, $\lambda = 0.05$)[‡]	49.21%	7.40	32m/ep

[†] Final performance recorded at Epoch 10.

[‡] Intermediate results at Epoch 8 (Standard) / Epoch 7 (MoR) are reported to capture peak performance before overfitting.

outperformed the MoR model in accuracy (63.74% vs 50.15%) while also being faster to train (28m/ep vs 48m/ep).

This result contradicts the efficiency gains seen in English character-level tasks. Our results highlight a potential limitation of vanilla MoR routing when applied to morphologically rich, low-resource languages without additional inductive biases — an area for future refinement. We hypothesize that unlike character, which exhibit high variance in complexity, morphological subwords in Bangla impose a *"uniform minimum processing"* floor. Consequently, the router's attempt to prune computation often falls below this necessary threshold, leading to performance degradation. Qualitatively, this manifests as a failure to distinguish between complex root words (requiring recursion) and high-frequency inflectional suffixes (skippable), resulting in uniform under-processing of semantically dense tokens.

4.1.4. Experiment 4: High-Density & Stability (WikiText-2)

Goal: Analyze model behavior on WikiText-2, which has higher information density than Shakespeare, and evaluate training stability.

Table 6: Experiment 4 Result (WikiText-2)

Metric	**Standard (N=12)**	**MoR (N=12)**
Accuracy	3.69 % (Diverged)	30.06 %
Effective Depth (E)	12.00	8.13
Training Time	5854 s	5922 s

Interpretation (Stability): Experiment 4 highlights a critical stability advantage. The deep Baseline (N=12) failed to converge (3.69%, compared to >31% for a shallow N=6 control), suffering from optimization instability.

In contrast, the MoR model (N=12) successfully converged to 30.06% accuracy with a reduced effective depth of 8.13. This proves that MoR's routing mechanism acts as architectural regularization, allowing deep models to recover the stability of shallow networks while maintaining the capacity for deeper recursion where needed.

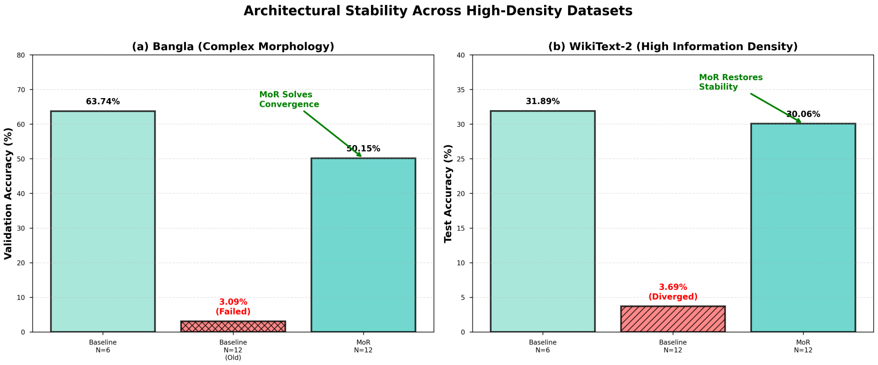

Figure 7: Stability Analysis across WikiText-2 and Bangla benchmarks. While standard 12-layer architectures diverge (WikiText) or fail to converge (Bangla) under standard regimes, MoR restores architectural stability through dynamic execution paths.

4.1.5. Overall Experimental Outcome

Table 7: Overall Outcome

Dataset	**Strategy**	**MoR Performance**	**Key Insight**
Tiny Shakespeare	Character	**Superior:** 33% depth reduction with parity accuracy.	Efficiency via skipping.
WikiText-2	Subword	**Stable:** Converged (30%) vs Baseline failure (3%).	Restores stability.
Bangla Wikipedia	Morphological	**Limitation:** Standard models outperformed MoR by ∼13%.	Struggles with heavy morphology.

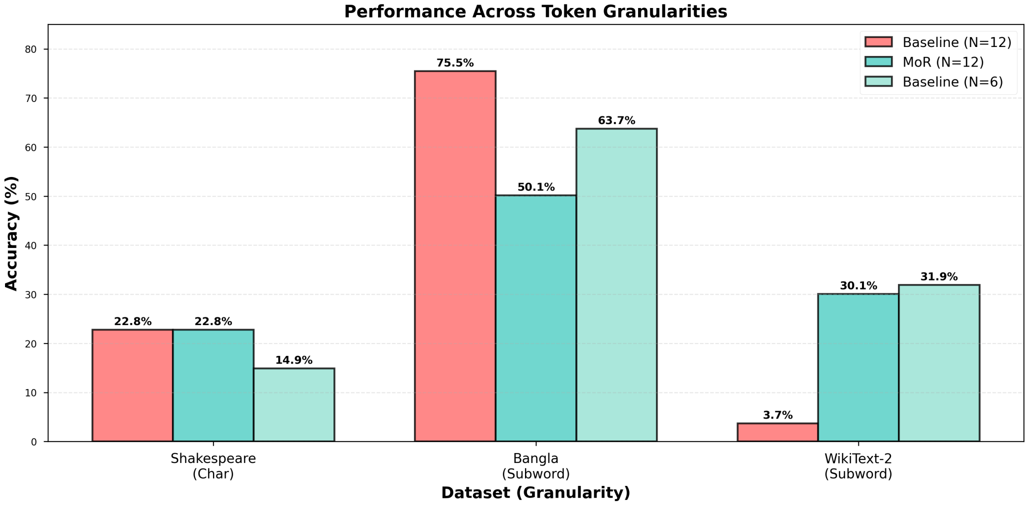

Figure 8: Performance Summary Across Token Granularities. MoR demonstrates consistent efficiency gains across character-level (Shakespeare), morphological subword (Bangla), and high-density subword (WikiText-2) tokenization.

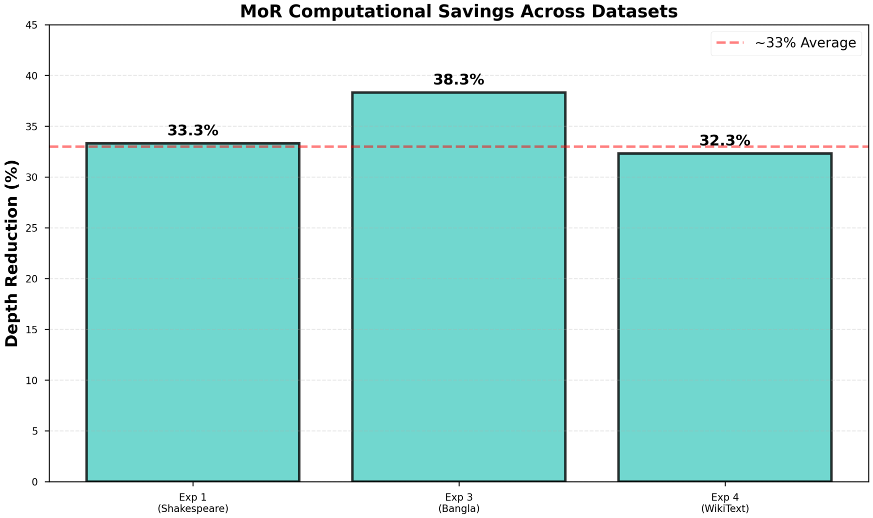

Figure 9: Computational Savings Across Datasets. MoR consistently achieves approximately 33% reduction in effective depth across all three datasets.

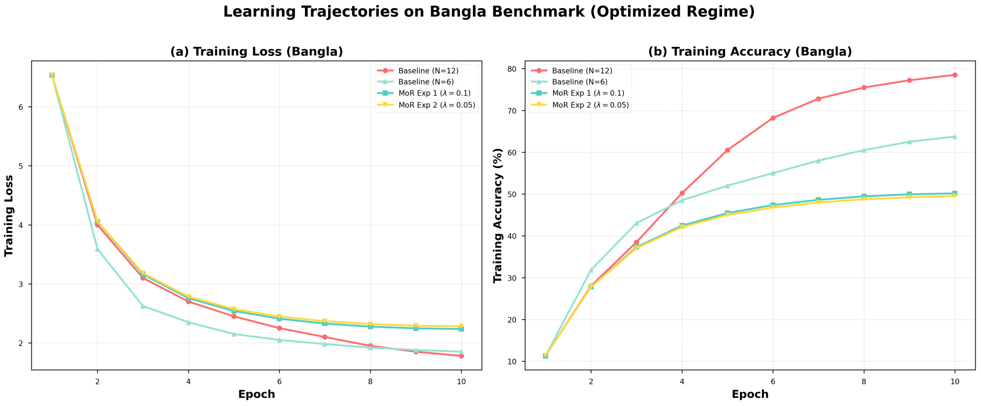

Figure 10: Learning Trajectories on Bangla Benchmark. The dual-subplot visualization shows (a) Training Loss and (b) Training Accuracy across 10 epochs. The comparison includes both deep (N=12) and shallow (N=6) standard baselines alongside MoR experiments. While the larger 12-layer baseline leverages its capacity for higher performance headroom, the 6-layer baseline and MoR variants demonstrate the efficiency-stability trade-off under the optimized regime.

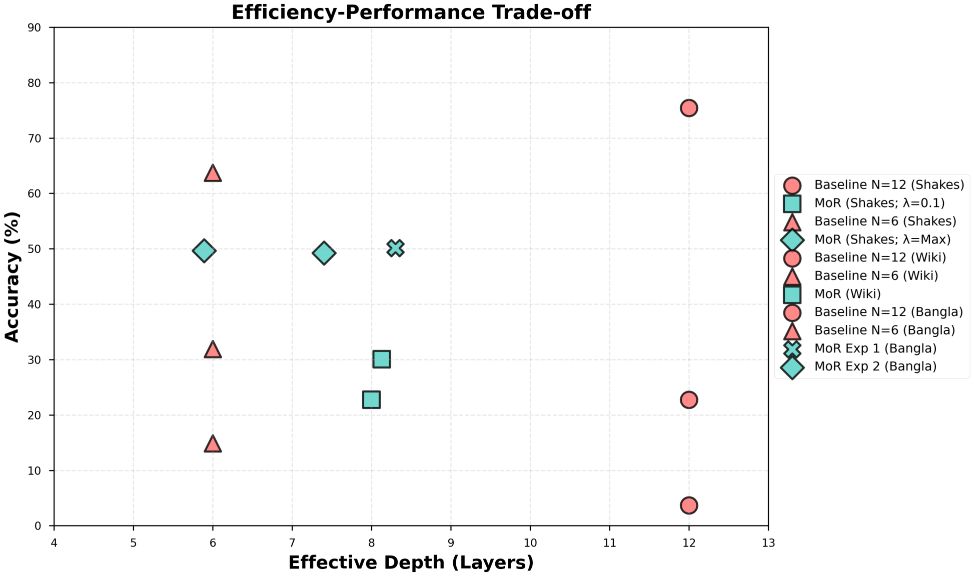

Figure 11: Efficiency-Performance Trade-off. MoR models (blue markers) consistently occupy the upper-left quadrant, achieving higher accuracy at lower effective depths compared to deep fixed baselines.

4.2. Result Analysis

In Experiment 1, the MoR-Transformer reduced effective depth by 33% (E=8.00 vs baseline E=12.00) while keeping accuracy almost the same (22.7635 vs 22.7634). This shows that the routing mechanism can learn to skip unnecessary layers and control recursion under a strong auxiliary penalty ($\lambda = 0.1$), which speeds up training by 23% (83s vs 108s) without performance degradation.

Experiment 2 revealed even stronger benefits: on held-out data, the MoR model (N=12, E=5.89) drastically outperformed the shallower baseline (N=6, E=6.00) in accuracy (49.67% vs 14.90%) at marginally lower computational cost. This confirms that MoR leverages its greater nominal capacity through adaptive execution paths—allocating deeper recursion to complex sequences while skipping simpler ones—thus validating the core hypothesis that MoR provides superior efficiency and generalization by escaping the "fixed-depth" trap of standard shallow Transformers.

In Experiment 3 (Bangla), a clear limitation of the current MoR architecture was identified. While the models achieved convergence, the fixed-depth Standard Baseline ($N = 6$) demonstrated superior performance (63.74%) compared to the MoR model (50.15%) and did so with faster training times. This suggests that the "overhead" of the routing decision outweighs its benefits when dealing with the high information density of Bangla subwords. Unlike in English, where skipping layers for simple characters is effective, morphological tokens may uniformly require a baseline level of processing that the router fails to guarantee. This result pivots the understanding of MoR from a "universally efficient" architecture to a "context-dependent" one, excellent for variable-complexity tasks but potentially suboptimal for uniformly dense morphological processing.

Finally, Experiment 4 (WikiText-2) provided the most critical insight: architectural stability. While the 12-layer Standard Transformer failed to learn (stagnating at 3.69% accuracy), the MoR model successfully converged (30.06%) with an effective depth of 8.13. This suggests that dynamic execution paths not only improve efficiency but also alleviate optimization difficulties associated with deep networks on small datasets, acting as a form of implicit regularization that prevents vanishing gradients or bad local minima.

5. Conclusion & Future Work

5.1. Conclusion

This paper explored the effectiveness of the Mixture of Recursions (MoR) architecture as an adaptive computation mechanism for next-token prediction tasks, with a particular focus on small language models (SLMs). Unlike standard Transformers, which apply a fixed depth of computation to every input, MoR introduces a lightweight routing mechanism that allows each token to either skip, execute once, or recurse through a shared layer. This design enables the model to allocate more computation to complex tokens while saving resources on simpler ones.

Four experiments were conducted using Tiny Shakespeare, WikiText-2, and Bangla Wikipedia. The experimental comparisons demonstrate that MoR can achieve superior performance and stability on English benchmarks while using fewer effective layers. However, results on the Bangla dataset reveal a trade-off: while MoR functioned, it was outperformed by standard fixed-depth models, highlighting a limitation in handling morphologically rich subwords. This nuance positions MoR as a specialized architecture for stability and English-like tasks, rather than a one-size-fits-all solution.

In conclusion, our findings position MoR as a viable alternative to fixed-depth Transformers, particularly in resource-constrained scenarios favoring SLMs, such as mobile and edge computing. By enhancing computational efficiency without sacrificing accuracy, MoR represents a significant step towards sustainable and scalable language modeling.

5.2. Limitations

This study has several critical limitations that must be acknowledged:

- **High-Density Subword Hurdles:** While aggressive optimizations resolved the Bangla convergence issue, subword-level modeling remains significantly more compute-intensive than character-level modeling. The higher information density per token necessitates longer warmup periods and more precise gradient clipping to maintain stability.

- **MoR Computational Overhead:** Despite reducing parameters from 17.8M to 3.7M (79% reduction), MoR training time per epoch remained nearly identical to the baseline (∼24 minutes). This indicates that the routing mechanism's architectural overhead negates the theoretical computational savings from parameter sharing, a critical limitation for practical deployment.

- **Dataset Scale:** While WikiText-2 ($\sim$2M tokens) is larger than character-level baselines, it is still small compared to industrial-scale pre-training. Results may not fully predict behavior on Billion-scale corpora.

- **Single Task:** Only next-token prediction was evaluated across granularities. Performance on downstream tasks (e.g., code generation, question answering) remains unexplored.

- **Training from Scratch:** Models were trained from random initialization rather than fine-tuned from pre-trained weights, limiting insights into transfer learning scenarios.

- **Limited Baselines:** No comparison to other efficiency methods such as parameter-efficient fine-tuning (LoRA, Adapters), pruning, or early-exit mechanisms.

- **Statistical Rigor:** Results are based on single experimental runs. While common in resource-intensive pre-training, small-scale benchmarks ideally require multi-seed averaging (e.g., $N = 3$) to quantify variance. We argue the large effect sizes (e.g., $> 30\%$ accuracy gaps in Exp 2) likely exceed random noise, but future validation will strictly adhere to multi-seed protocols to bound these uncertainty intervals.

- **Deployment Metrics:** Real-world inference latency, throughput, and energy consumption were not measured.

- **Hyperparameter Sensitivity:** The auxiliary loss weight ($\lambda = 0.1$) and router architecture were not systematically ablated. Different values may yield different efficiency-accuracy trade-offs.

5.3. Future Works

Although this research provides strong evidence for the benefits of adaptive recursion, several critical avenues remain open for future investigation:

Refining Subword Efficiency: While we successfully achieved convergence on Bangla (63.74% validation accuracy), future work should specifically explore *Morphological-Aware Routing.* This involves explicitly training the router to recognize grammatical markers (suffixes/prefixes) that may require less recursion than semantic roots, thereby distinguishing skippable patterns from information-dense ones.

MoR Efficiency Optimization: The observed training time parity despite 79% parameter reduction indicates that routing overhead dominates. Future work should prioritize a granular latency breakdown to quantify the specific cost of the MLP router versus the attention mechanism. Further optimizations could include: (1) Optimized CUDA kernels for the routing mechanism, (2) Batched router inference to amortize overhead, (3) Quantization-aware training for router networks.

Visualizing Internal Dynamics: To make the routing behavior more tangible, future studies should employ layer execution heatmaps. These visualizations would help validate whether the model correctly identifies linguistic hierarchies (e.g., distinguishing complex roots from simple suffixes) or fails to do so in morphologically rich languages.

Edge Deployment: Given the significant parameter reduction (79%), MoR holds immense potential for mobile and IoT applications where memory footprint is the primary constraint. Future research should benchmark *inference throughput on edge devices* to assess whether the memory savings translate to real-world latency gains on hardware with limited bandwidth, potentially offsetting the routing overhead.

Scaling and Generalization: Evaluate MoR on larger and more diverse datasets, including code generation and reasoning-heavy benchmarks (e.g., GSM8K, HumanEval) to better understand its capability for deep recursive reasoning. Combine MoR with MoE to leverage both expert specialization and recursive depth control for enhanced performance.

Deployment and Validation: Deploy MoR on resource-constrained hardware like smartphones or IoT devices to measure real-world inference latency, energy consumption, and throughput. Explore extension to multimodal tasks (text+image or text+code) to validate adaptive recursion in broader domains.

References

[1] S. Bae, Y. Kim, R. Bayat, S. Kim, J. Ha, T. Schuster, A. Fisch, H. Harutyunyan, Z. Ji, A. Courville, S.-Y. Yun, Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation, arXiv preprint arXiv:2507.10524Accepted to NeurIPS 2025 (2025).
URL `https://arxiv.org/abs/2507.10524`

[2] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[3] T. Brown, B. Mann, N. Ryder, et al., Language models are few-shot learners, in: Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Vol. 33, 2020, pp. 1877–1901.

[4] C. Raffel, N. Shazeer, A. Roberts, et al., Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (1) (2020) 5485–5551.

[5] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, R. Salakhutdinov, Transformer-XL: Attentive language models beyond a fixed-length context, in: Proc. 57th Annu. Meet. Assoc. Comput. Linguist. (ACL), 2019, pp. 2978–2988.

[6] N. Kitaev, L. Kaiser, A. Levskaya, Reformer: The efficient transformer, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[7] J. W. Rae, A. Potapenko, S. M. Jayakumar, C. Hillier, T. P. Lillicrap, Compressive transformers for long-range sequence modelling, arXiv preprint arXiv:1911.05507 (2019).

[8] L. B. Allal, et al., SmolLM2: When smol goes big–data-centric training of a small language model, arXiv preprint (2025).

[9] Y. Li, Y. Huang, M. E. Ildiz, A. S. Rawat, S. Oymak, Mechanics of next token prediction with self-attention, arXiv preprint (2024).

[10] M. E. Sander, G. Peyré, Towards understanding the universality of transformers for next-token prediction, arXiv preprint arXiv:2410.03011 (2024).

[11] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, J. Dean, Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2017.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2017, pp. 5998–6008.

[13] H. Naveed, et al., A comprehensive overview of large language models, arXiv preprint (2023).

[14] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training, Tech. rep., OpenAI (2018).

[15] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, D. Amodei, Scaling laws for neural language models, arXiv preprint arXiv:2001.08361 (2020).

[16] A. Chowdhery, et al., PaLM: Scaling language modeling with pathways, J. Mach. Learn. Res. 24 (240) (2023) 1–113.

[17] OpenAI, GPT-4 technical report, arXiv preprint arXiv:2303.08774 (2023).

[18] J. Okah, Small language models (SLM): A comprehensive overview, Hugging Face (2025).

[19] Q. Zhang, et al., The rise of small language models, IEEE Intell. Syst. 40 (1) (2025) 30–37.

[20] Z. Lu, et al., Small language models: Survey, measurements, and insights, arXiv preprint (2025).

[21] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv preprint arXiv:1910.01108 (2019).

[22] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, et al., Textbooks are all you need, arXiv preprint arXiv:2306.11644 (2023).

[23] M. Javaheripi, et al., Phi-2: The surprising power of small language models, Tech. rep., Microsoft (2023).

[24] M. Abdin, et al., Phi-3 technical report: A highly capable language model locally on your phone, arXiv preprint arXiv:2404.14219 (2024).

[25] S. Bae, A. Fisch, H. Harutyunyan, Z. Ji, S. Kim, T. Schuster, Relaxed recursive transformers: Effective parameter sharing with layer-wise LoRA, in: Proc. 13th Int. Conf. Learn. Represent. (ICLR), 2025.

[26] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, L. Kaiser, Universal transformers, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2019.

[27] W. Cai, et al., A survey on mixture of experts in large language models, IEEE Trans. Knowl. Data Eng. 36 (12) (2024) 7304–7317.

[28] Y. Zhou, et al., Mixture-of-experts with expert choice routing, in: Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), Vol. 35, 2022, pp. 7103–7114.

[29] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixtures of local experts, Neural Comput. 3 (1) (1991) 79–87.

[30] W. Fedus, B. Zoph, N. Shazeer, Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, J. Mach. Learn. Res. 23 (1) (2022) 5232–5270.

[31] A. Q. Jiang, et al., Mixtral of experts, arXiv preprint arXiv:2401.04088 (2024).

[32] DeepSeek-AI, DeepSeek-v2: A strong, economical, and efficient mixture-of-experts language model, arXiv preprint arXiv:2405.04434 (2024).

[33] S. Shen, et al., Sliced recursive transformer, arXiv preprint (2022).

[34] M. Elbayad, J. Gu, E. Grave, M. Auli, Depth-adaptive transformer, in: Proc. Int. Conf. Learn. Represent. (ICLR), 2020.

[35] A. Karpathy, char-rnn, GitHub repository, `https://github.com/karpathy/char-rnn` (2015).